

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO ĐỒ ÁN**

**CẤU TRÚC DỮ LIỆU & GIẢI THUẬT**

**Ứng dụng ngăn xếp tạo Trò chơi Tháp Hà Nội**

**SVTH : NGUYỄN THÀNH NHƯ**

**MSSV : 17110202**

**Tp. Hồ Chí Minh, tháng 12** **năm 2018**

**MỤC LỤC**

[LỜI NÓI ĐẦU 1](#_Toc530910410)

[I. Lí do chọn đề tài 2](#_Toc530910411)

[II. Mục tiêu cho đề tài 2](#_Toc530910412)

[III. Phương pháp nghiên cứu 2](#_Toc530910413)

[IV. Giới thiệu đề tài 2](#_Toc530910414)

[V. Mô tả ứng dụng 4](#_Toc530910415)

[1. Chức năng phần mềm 4](#_Toc530910416)

[2. Use case diagram của phần mềm 4](#_Toc530910417)

[3. Mô tả ý nghĩa của từng use case 4](#_Toc530910418)

[3.1. Use case BtnShowRule 4](#_Toc530910419)

[3.2. Use case tmCoutTime 4](#_Toc530910420)

[3.3. Use case lbmoveCount 5](#_Toc530910421)

[3.4. Use case ProcessMovingDisk, MoveDisk 5](#_Toc530910422)

[3.5. Use case BtnPlay 5](#_Toc530910423)

[3.6. Use case BtnGiveIn 5](#_Toc530910424)

[3.7. Use case PicRod\_Click 5](#_Toc530910425)

[VI. Mô tả quá trình thực hiện 5](#_Toc530910426)

[1. Thiết kế giao diện 5](#_Toc530910427)

[1.1. Chuẩn bị hình ảnh 5](#_Toc530910428)

[1.2. Thiết kế giao diện với Windows Forms Application 6](#_Toc530910429)

[2. Ứng dụng kiểu dữ liệu Stack trong Trò chơi Tháp Hà Nội 7](#_Toc530910430)

[2.1. Các phương thức được sử dụng với kiểu dữ liệu Ngăn xếp (Stack) trong đề tài: 8](#_Toc530910431)

[2.2. Các *Stack* trong đề tài: 8](#_Toc530910432)

[2.3. Sử dụng các Stack trong đề tài: 9](#_Toc530910433)

[3. Viết code phần xử lí 10](#_Toc530910434)

[3.1. Nút “Chịu Thua” 10](#_Toc530910435)

[3.2. Nút “Chơi” 10](#_Toc530910436)

[3.3. Hàm xử lí Click Cột 11](#_Toc530910437)

[3.4. Hàm xử lí quá trình di chuyển đĩa 12](#_Toc530910438)

[3.5. Hàm di chuyển đĩa 13](#_Toc530910439)

[4. Chạy và kiểm thử 15](#_Toc530910440)

[5. Sửa các lỗi gặp phải 17](#_Toc530910441)

[VII. Mô tả phân công công việc 17](#_Toc530910442)

[VIII. Kết luận 17](#_Toc530910443)

[IX. Tài liệu tham khảo 18](#_Toc530910444)

**MỤC LỤC HÌNH**

[Hình 1: Mô hình Tháp Hà Nội trong thực tế. 3](#_Toc530684485)

[Hình 2: Use case diagram GameThapHaNoi. 4](#_Toc530684486)

[Hình 3: Hình ảnh các đĩa và cột được chuẩn bị. 6](#_Toc530684487)

[Hình 4: Khởi tạo new project với Visual Studio. 6](#_Toc530684488)

[Hình 5: Giao diện cơ bản sau khi thiết kế. 7](#_Toc530684489)

[Hình 6: Hàm xử lí nút “Chịu Thua”. 10](#_Toc530684490)

[Hình 7: Hàm xử lí Nút “Chơi”. 10](#_Toc530684491)

[Hình 8: Hàm xử lí click Cột 12](#_Toc530684492)

[Hình 9: Hàm xử lí quá trình di chuyển đĩa. 13](#_Toc530684493)

[Hình 10: Hàm xử lí di chuyển đĩa. 14](#_Toc530684494)

[Hình 11: Trò chơi Tháp Hà Nội khi vừa mở lên. 15](#_Toc530684495)

[Hình 12: Trò chơi Tháp Hà Nội khi vừa chọn chế độ chơi. 15](#_Toc530684496)

[Hình 13: Tiến hành di chuyển đĩa. 16](#_Toc530684497)

[Hình 14: Thông báo chúc mừng khi người chơi hoàn thành trò chơi. 16](#_Toc530684498)

[Hình 15: Hàm xử lí click đĩa. 17](#_Toc530684499)

# LỜI NÓI ĐẦU

Cấu trúc dữ liệu và giải thuật là một trong những môn học nền tảng quan trọng nhất dành cho sinh viên ngành Công nghệ Thông tin. Môn học cung cấp cho sinh viên hiểu và biết cách sử dụng các kiểu dữ liệu cơ bản thường gặp trong quá học tập và làm việc. Giúp người học biết được sự tác động lẫn nhau giữa dữ liệu và giải thuật để người sử dụng có thể giải quyết các vấn đề gặp phải trong bài toán và thực tế.

Sau quá trình học tác giả đã vận dụng kiến thức tiếp thu được để ứng dụng thực hiện đồ án môn học với đề tài: Ứng dụng ngăn xếp tạo Trò chơi Tháp Hà Nội.

# Lí do chọn đề tài

Với đề tài “Ứng dụng ngăn xếp tạo Trò chơi Tháp Hà Nội” có thể giúp tác giả vận dụng những kiến thức học được từ môn học, đặc biệt là hiểu rõ hơn về kiểu dữ liệu trừu trượng Stack. Ngoài ra việc tạo ứng dụng Trò chơi Tháp Hà Nội sẽ giúp người chơi có thể chơi được trò chơi mà không cần mô hình thực tế, phục vụ cho nhu cầu giải trí của người chơi.

# Mục tiêu cho đề tài

Hoàn thành sản phẩm, có thể cho người chơi được, trải nghiệm trò chơi thành công. Giúp bản thân tác giả hiểu được, hiểu sâu hơn về kiểu dữ liệu trừu tượng ngăn xếp (Stack). Có thể sử dụng thành thạo các phương thức và thuộc tính của Stack.

# Phương pháp nghiên cứu

Vận dụng các kiến thức đã học được từ bộ môn “Cấu trúc dữ liệu và giải thuật”, tham khảo các nguồn tài liệu, giáo trình được cung cấp. Ngoài ra còn tham khảo thêm các nguồn tài liệu khác trên mạng internet và vận dụng tất cả để thực hiện đề tài nghiên cứu một cách thành công nhất.

# Giới thiệu đề tài

Trò chơi (Bài toán) Tháp Hà Nội có thể đã xuất hiện ở Đông Á từ thế kỷ 19 hoặc trước đó, trò chơi được phổ biến rộng rãi ở Paris năm 1883 bởi nhà toán học Edouard Lucas, là một bài toán nổi tiếng thế giới, hiện nay đang được nghiên cứu bởi rất nhiều nhà toán học và khoa học máy tính, được đưa vào nhiều giáo trình tin học và trò chơi toán học như một ví dụ điển hình về thuật toán đệ qui và lập trình căn bản. [1]

Về luật chơi người chơi được cho ba cái cọc và một số đĩa có kích thước khác nhau có thể cho vào các cọc này. Ban đầu, sắp xếp các đĩa theo trật tự kích thước vào một cọc sao cho đĩa nhỏ nhất nằm trên cùng. Tức là tạo ra một dạng hình nón. Người chơi phải di chuyển toàn bộ số đĩa sang một cọc khác, tuân theo các quy tắc sau: [2]  
 + Mỗi lần chỉ được di chuyển một đĩa.

+ Mỗi đĩa chỉ có thể được đặt lên một đĩa lớn hơn.



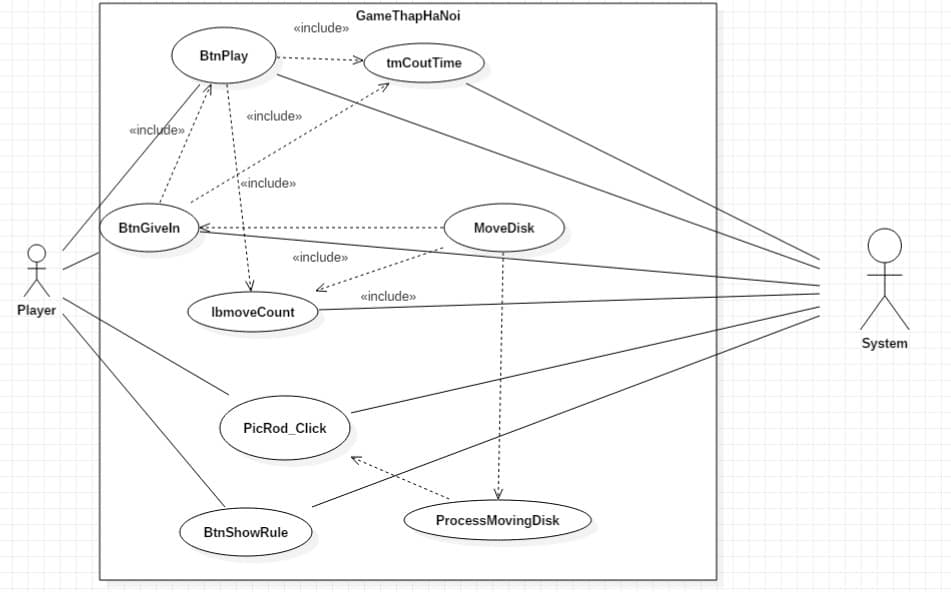
Hình 1: Mô hình Tháp Hà Nội trong thực tế.

# Mô tả ứng dụng

## Chức năng phần mềm

Mô phỏng trò chơi Tháp Hà Nội thành ứng dụng để người chơi có thể chơi mà không cần có mô hình thực tế. Ứng dụng trong giải trí, rèn luyện tư duy logic, trong học tập và giảng dạy về thuật toán đệ qui và lập trình căn bản.

## Use case diagram của phần mềm



Hình 2: Use case diagram GameThapHaNoi.

## Mô tả ý nghĩa của từng use case

### Use case BtnShowRule

* Mục đích: biểu diễn luật chơi
* Tình huống sử dụng: khi người chơi cần biết về luật chơi.

### Use case tmCoutTime

* Mục đích: hiển thị thời gian chơi ra giao diện.
* Tình huống sử dụng: khi người chơi bắt đầu tham gia trò chơi và dừng khi chiến thắng hoặc người chơi chịu thua.

### Use case lbmoveCount

* Mục đích: hiển thị số lần di chuyển ra giao diện.
* Tình huống sử dụng: khi người chơi di chuyển 1 đĩa từ cột này sang cột kia.

### Use case ProcessMovingDisk, MoveDisk

* Mục đích: thực hiện việc di chuyển đĩa.
* Tình huống sử dụng: giúp người chơi di chuyển đĩa từ cột này sang cột khác.

### Use case BtnPlay

* Mục đích: cho người chơi bắt đầu chơi khi click.
* Tình huống sử dụng: khi người chơi muốn bắt đầu trò chơi và click vào nút.

### Use case BtnGiveIn

* Mục đích: cho người sử dụng khi muốn chịu thua.
* Tình huống sử dụng: khi người chơi muốn chịu thua, dừng trò chơi.

### Use case PicRod\_Click

* Mục đích: cho người chơi thao tác click các đĩa, các cột.
* Tình huống sử dụng: khi người chơi muốn click để chọn, hoặc bỏ chọn đĩa, cột để di chuyển.

# Mô tả quá trình thực hiện

## Thiết kế giao diện

Việc thiết kế giao diện của ứng dụng là một phần rất quan trọng trong việc xây dựng ứng dụng, là bộ mặt của ứng dụng, nơi người dùng trực tiếp thao tác với ứng dụng.

### Chuẩn bị hình ảnh

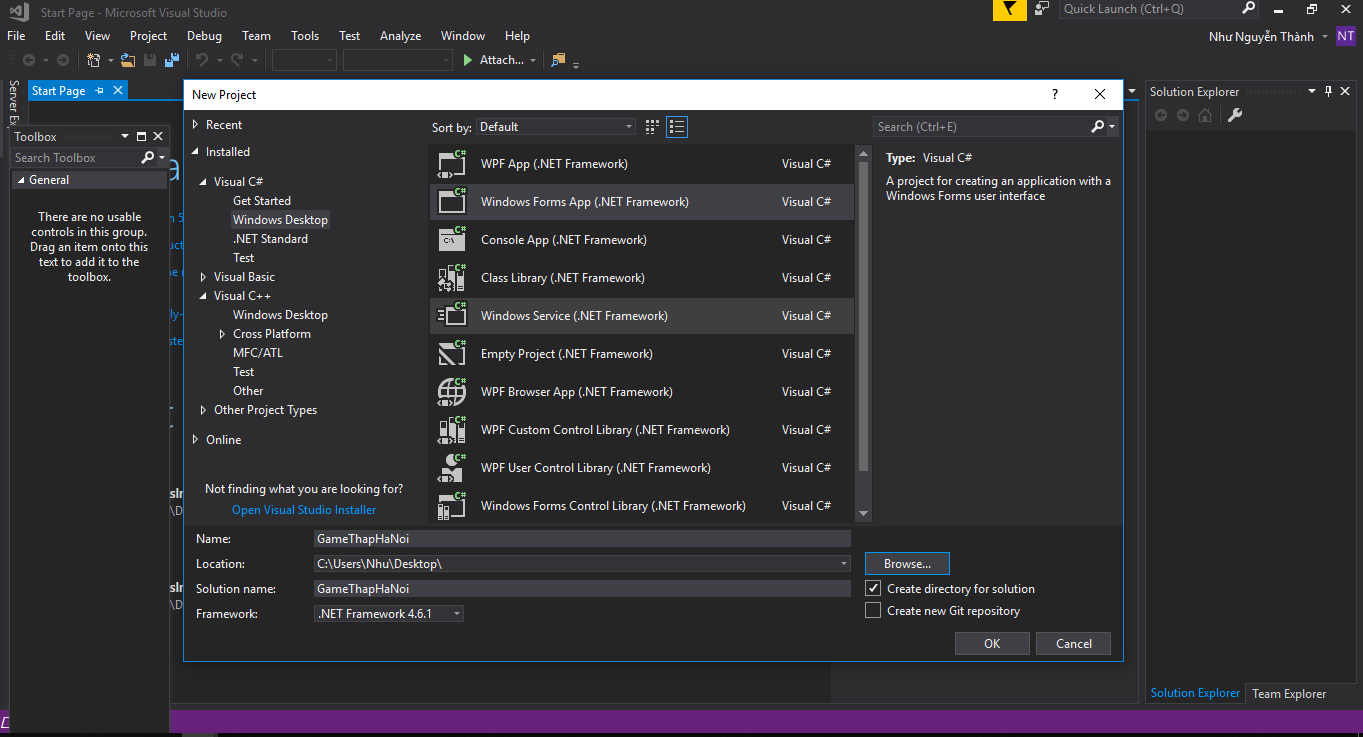


Hình 3: Hình ảnh các đĩa và cột được chuẩn bị.

* Vẽ đĩa: vẽ 8 đĩa với các màu sắc, kích thước khác nhau được đánh số thứ tự dựa theo kích thước của tững đĩa.
* Vẽ cột: vẽ 3 cột với màu sắc phù hợp, có hình dáng mô phỏng cột tháp Hà Nội.
* Tùy chỉnh size ảnh phù hợp: đây là một phần rất quan trọng, các đĩa phải có cùng chiều cao và size chiều rộng (cụ thể 330x27), tương tự các cột cũng như vậy (cụ thể 353x325). Việc tạo size ảnh phù hợp sẽ giúp việc lập trình các hàm sử lí được đơn giản, có tính thống nhất và dễ dàng hơn rất nhiều vì thế chúng ta nên tùy chỉnh một cách phù hợp nhất có thể.

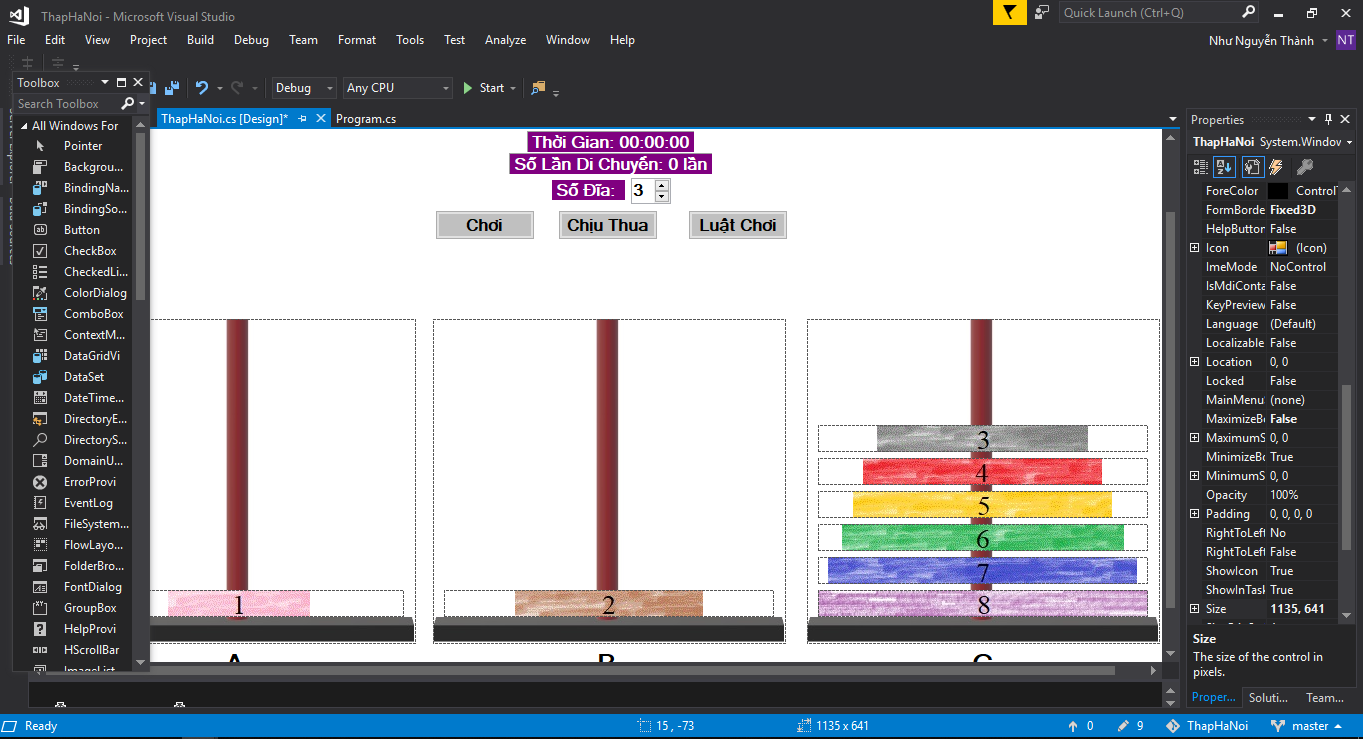
### Thiết kế giao diện với Windows Forms Application

Kể từ lúc này chúng ta sẽ bắt đầu sử dụng Windows Forms Application để thiết kế và lập trình ứng dụng.



Hình 4: Khởi tạo new project với Visual Studio.

* Sau khi tạo project thành công, chúng ta bắt đầu điều chỉnh form với các thao tác trên thanh properties. Chúng ta sẽ thay đổi màu nền (*Backcolor*), cố định size không cho tùy chỉnh (*MaximizeBoxFalse*), chỉnh lại mục *Name* và mục *Text*.
* Thêm hình ảnh trụ và đĩa từ thư mục vào *Project Resources*.
* Tạo các *PictureBox* trong thanh *Toolbox* để hiển thị các cột, đĩa (khi mới vô trò chơi các đĩa không hiển thị nên ta điều chỉnh thuộc tính *Visible: False*) sau đó tạo thêm *label* ghi chú thứ tự các cọc (A, B, C).
* Tiến hành tinh chỉnh vị trí, canh giữa các cột so với form, canh giữa các đĩa so với cột mà đĩa đang nằm. Điều chỉnh căn giữa vị trí các *label* thứ tự cọc A, B, C so với từng cột.
* Tạo *label* đếm thời gian (sử dụng *timer* trong thanh *Toolbox*) và *label* đếm số lần di chuyển.
* Tạo *NumericUpDown* cho người chơi chọn số lượng đĩa (level trò chơi, với tùy chình level thấp nhất là 2 và cao nhất là 8).
* Tạo 3 nút (*Button*) “Chơi”, “Chịu Thua” và “Luật Chơi”, căn giữa điều chỉnh vị trí phù hợp so với form.



Hình 5: Giao diện cơ bản sau khi thiết kế.

## Ứng dụng kiểu dữ liệu Stack trong Trò chơi Tháp Hà Nội

Ngăn xếp (Stack) là một dạng danh sách được cài đặt nhằm sử dụng cho các ứng dụng cần xử lý theo thứ tự đảo ngược. Trong cấu trúc dữ liệu ngăn xếp, tất cả các thao tác thêm, xóa một phần tử đều phải thực hiện ở đầu một danh sách, đầu này gọi là đỉnh (top) của ngăn xếp. Hoạt động dựa trên quy tắc LIFO (Last In, First Out). Tương tự như ngăn xếp, trong trò chơi Tháp Hà Nội mỗi thao tác thực hiện di chuyển đĩa của chúng ta đều chỉ thực hiện với đĩa trên cùng của cột do đó chúng ta sẽ sử dụng được kiểu dữ liệu Stack để lưu giá trị các đĩa trong từng cột là cách tối ưu và hiệu quả nhất.

### Các phương thức được sử dụng với kiểu dữ liệu Ngăn xếp (Stack) trong đề tài:

* *Push()*: thêm đĩa vào đầu cột.
* *Pop():* lấy phần tử đầu ra khỏi cột.
* *Peek()*: lấy thông tin phần tử đầu ra khỏi cột.
* *Contai*ns(): tìm xem trong *Stack* có chứa đĩa nào hay không.
* *Count:* kiểm tra xem *Stack* đang có bao nhiêu đĩa.

### Các *Stack* trong đề tài:



Hình 6: Các Stack được sử dụng trong đề tài.

* Các *Stack* chứa các đĩa lưu ở dạng *PictureBox.*
* *disksA, disksB, disksC* tương ứng với các cột đĩa trong trò chơi.
* *firstClickedDisks, secondClickedDisks* là hai *Stack* chứa các đĩa ở lần click thứ nhất và lần click thứ 2.

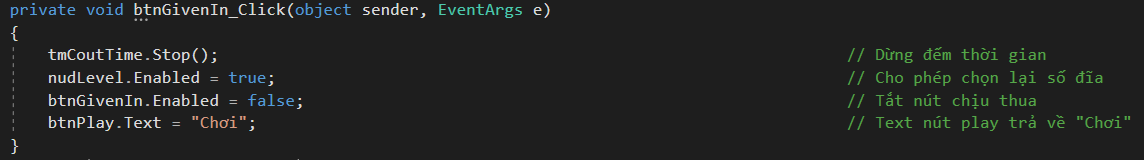
### Sử dụng các Stack trong đề tài:

* *Stack* trong nút “Chơi”, khi click vào nút chơi thì xảy ra 2 trường hợp đó là trò chơi đang diễn ra thì chúng ta phải trả các giá trị của các *Stack disksA, disksB, disksC* thì ta phải xóa các Stack trở về trạng thái rỗng bằng cách gọi lệnh *disksA.Clear(),* ngoài ra *firstClickedDisks, secondClickedDisks* cũng sẽ được trả về giá trị *null.* Trong trường hợp còn lại thì chúng ta chỉ sử dụng để thêm các đĩa vào cột A bằng cách gọi lệnh *disksA.Push(disks[i]).*
* Trong hàm xử lí Click cột chúng ta gọi mới một *Stack disksOfClickedRod* để lấy ra vá gán giá trị cột đĩa vừa được click vào nó. Trong trường hợp *firstClickedDisks == null* thì ta sẽ kiểm tra *disksOfClickedRod.Count == 0* nếu đúng thì có nghĩa cột vừa click không có đĩa nên ta sẽ không xử lí lần click này ngược lại thì ta lưu giá trị *firstClickedDisks = disksOfClickedRod.* Nếu lần click đầu khác *null* và lần click sau bằng *null* thì ta sẽ tiến hành kiểm tra xem lần click 2 có trùng với lần click 1 hay không (*disksOfClickedRod== firstClickedDisks*), nếu trùng ta trả giá trị lần click 1 về *null* ngược lại thì *secondClickedDisks = disksOfClickedRod.*
* Trong hàm xử lí quá trình di chuyển đĩa Stack tiếp tục được sử dụng. Trong hàm này chúng ta tiến hành lấy thông tin đĩa đầu của lần click đầu và lần click sau (gọi lệnh *firstClickedDisks.Peek()* và *secondClickedDisks()*) để so sánh giá trị hai đĩa có hợp lệ hay không để phục vụ cho việc di chuyển đĩa. Nếu không hợp lệ (đĩa lần click đầu lớn hơn) thì không tiến hành di chuyển đĩa, trả giá trị *Stack* *secondClickedDisks = null.*
* Với hàm di chuyển đĩa đầu tiên chúng ta sẽ lấy đĩa đầu của cột được chọn bằng lệnh *firstClickedDisks.Pop(),* sau đó tiến hành thêm đĩa vừa lấy ra vào cột được chọn sau với lệnh *secondClickedDisks.Pop().* Tiếp theo ta trả giá trị *firstClickedDisks, secondClickedDisks* về *null.* Cuối cùng chúng ta kiểm tra số đĩa có trong cột C nếu bằng số đĩa đã chọn thì thông báo chiến thắng (*disksC.Count == nudLevel.Value*).

## Viết code phần xử lí

### Nút “Chịu Thua”

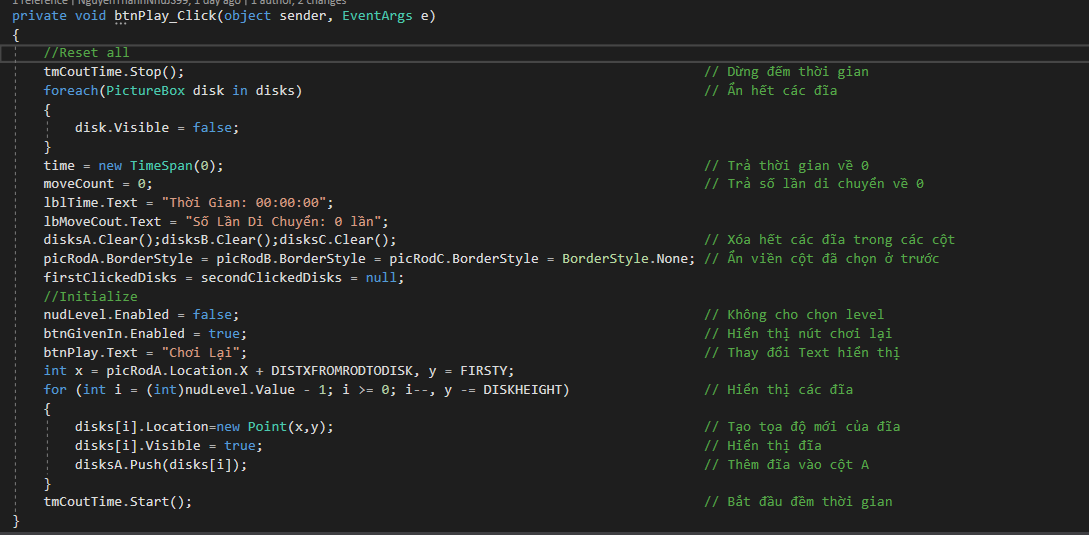
Khi người chơi click vào nút “Chịu Thua” lúc đó đồng hồ đếm thời gian sẽ dừng lại, nút “Chịu Thua” sẽ được tắt đi, *NumericUpDown* được mở lên và Text của BtnPlay sẽ hiển thị thông tin là “Chơi Lại”.



Hình 7: Hàm xử lí nút “Chịu Thua”.

### Nút “Chơi”

Với nút “Chơi” có trường hợp để xử lí. Một là khi trò chơi đang diễn chúng ta phải trả lại các giá trị về ban đầu, trường hợp hai là chưa bắt đầu chúng ta sẽ khởi tạo lại mới.



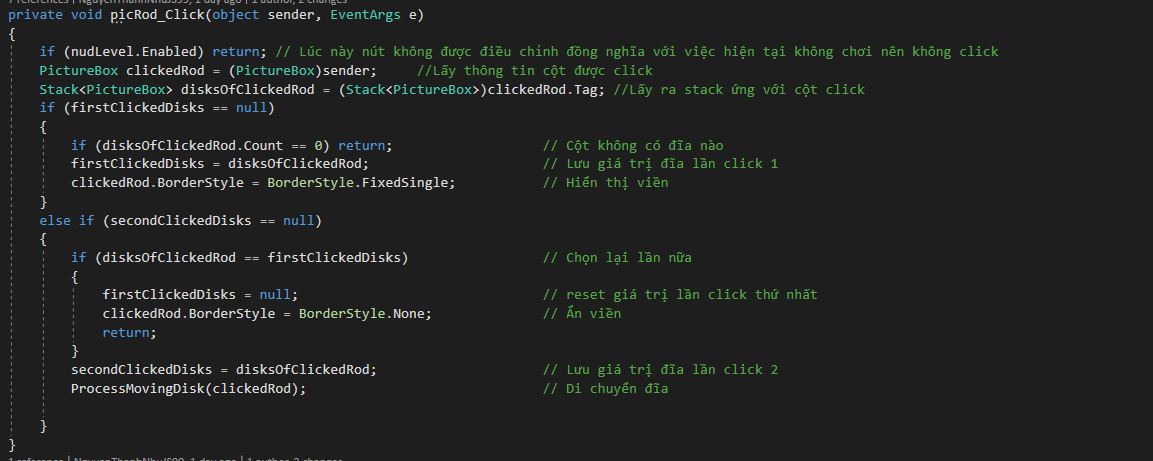
Hình 8: Hàm xử lí Nút “Chơi”.

* Trường hợp 1 khi trò chơi còn diễn ra: gọi lệnh *tmCoutTime.Stop()* để dừng đếm thời gian, thực hiện ẩn hết các đĩa bằng cách sử dụng *foreach* duyệt ẩn hiển thị hết của các đĩa trong Stack disk (disk là một stack chứa tất cả các đĩa), trả các giá trị Text của thời gian và số lần di chuyển về ban đầu. Sau đó tiến hành xóa các đĩa trong các cọc bằng lệnh *diskA.Clear()* và tiếp đến là ẩn các viền của cả 3 cột bằng cách trả thuộc tính *BoderStyle* của từng cột về *null.*
* Khi khởi tạo mới trò chơi bắt đầu thì bắt buộc chúng ta phải không cho người chơi thay đổi số đĩa lúc trò chơi đang diễn ra bằng trả thuộc tính *Enabled* của nó về *false,* và hiển thị nút chơi lại khi trả thuộc tính *Enabled* của nó về *true.* Tiếp đến là thao tác hiển thị các đĩa ở cột A lên bằng cách sử dụng vòng *for* với tọa độ x cố định và y giảm dần theo chiều cao của đĩa. Sau cùng chúng ta khởi động lại hàm đếm thời gian nhờ lệnh *tmCoutTime.Start().*

### Hàm xử lí Click Cột

Khi click cột sẽ được chia là 3 trường hợp. Trường hợp đầu là khi chưa chơi thì sẽ không click được, do đó trong trường hợp này chúng ta chỉ cần kiểm tra *NumericUpDown* đang bằng giá trị *false* thì sẽ không cho xử lí. Hai trường hợp còn lại là đó là lần click cột thứ nhất hoặc đó là lần click cột thứ hai khác cột đầu. Và để thực hiện thì đầu tiên ta phải lấy thông tin cột được click đưa về dạng *Stack<PictureBox>.*

* Lần click đầu: kiểm tra số đĩa trong cột được click nếu số đĩa trong cột bằng 0 (kiểm tra thuộc tính *Count*) thì sẽ không hiển thị viền của cột. Ngược lại thì ta lưu giá trị của cột được click vào *Stack firstClickedDisks* (*firstClickedDisks = disksOfClickedRod*) và sau đó hiển thị viền của cột vừa click.
* Lần click sau: trong trường hợp cột được click ở lần 2 trùng với cột được click ở lần 1 ta sẽ tiến hành bỏ chọn cột lần 1 đó, trả giá trị *firstClickedDisks* về *null,* sau đó tiến hành ẩn đi viền của cột đã chọn. Trong trường hợp ngược lại ta tiến hành lưu giá trị lần click thứ 2 sau đó tiến hành di chuyển đĩa bằng hàm *ProcessMovingDisk.*

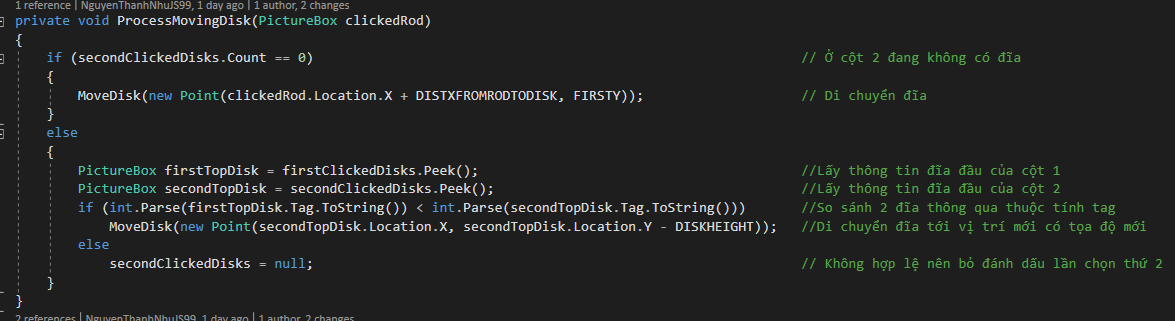


Hình 9: Hàm xử lí click Cột

### Hàm xử lí quá trình di chuyển đĩa

Trong quá trình xử lí này cũng xảy ra hai trường hợp để chúng ta phải xử lí. Trường hợp 1 là trong cột thứ 2 được chọn chưa có đĩa nào và trường hợp 2 là đã có đĩa trong cột.

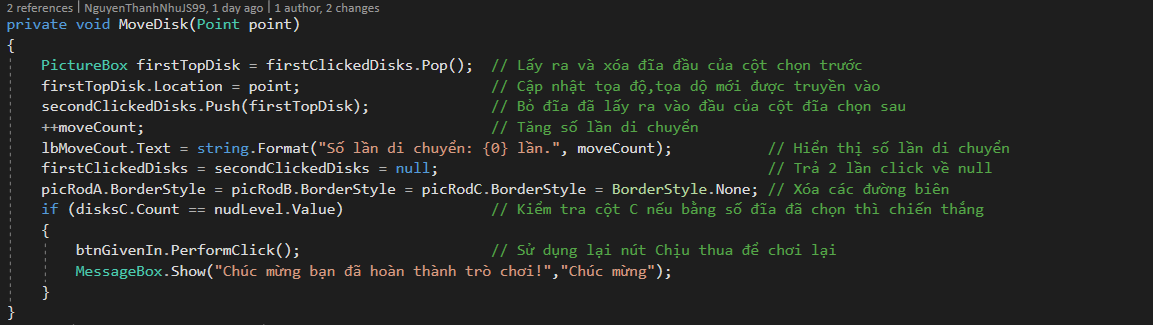
* Trong trường hợp 1: khi trong cột chưa có đĩa nào ta thực hiện gọi hàm xử lí *MoveDisk()* với tọa độ x là tọa độ của cột được chọn cộng với DISTXFROMRODTODISK và tọa độ y là tọa độ FIRSTY.
* Trong trường hợp 2: đầu tiên là lấy thông tin đĩa đầu của cột thứ nhất bằng cách gọi *PictureBox firstTopDisk* = *firstClickedDisks.Peek() với* phương thức *Peek()* của *Stack*, tương tự ta cũng sẽ lấy thông tin đĩa đầu của cột thứ hai. Sau đó so sánh giá trị của hai đĩa đầu của hai cột (thông qua thuộc tính Tag). Nếu giá trị đĩa đầu cột hai lớn hơn thì ta sẽ tiến hành di chuyển đĩa bằng hàm xử lí *MoveDisk()* với tọa độ x là tọa độ của đĩa trên cùng cột 2 và tọa độ y là tọa độ của đĩa trên cùng cột 2 trừ đi cho chiều cao của đĩa. Ngược lại nếu giá trị không phù hợp tiến hành bỏ chọn cột thứ 2 bằng lệnh *secondClickedDisks = null*.



Hình 10: Hàm xử lí quá trình di chuyển đĩa.

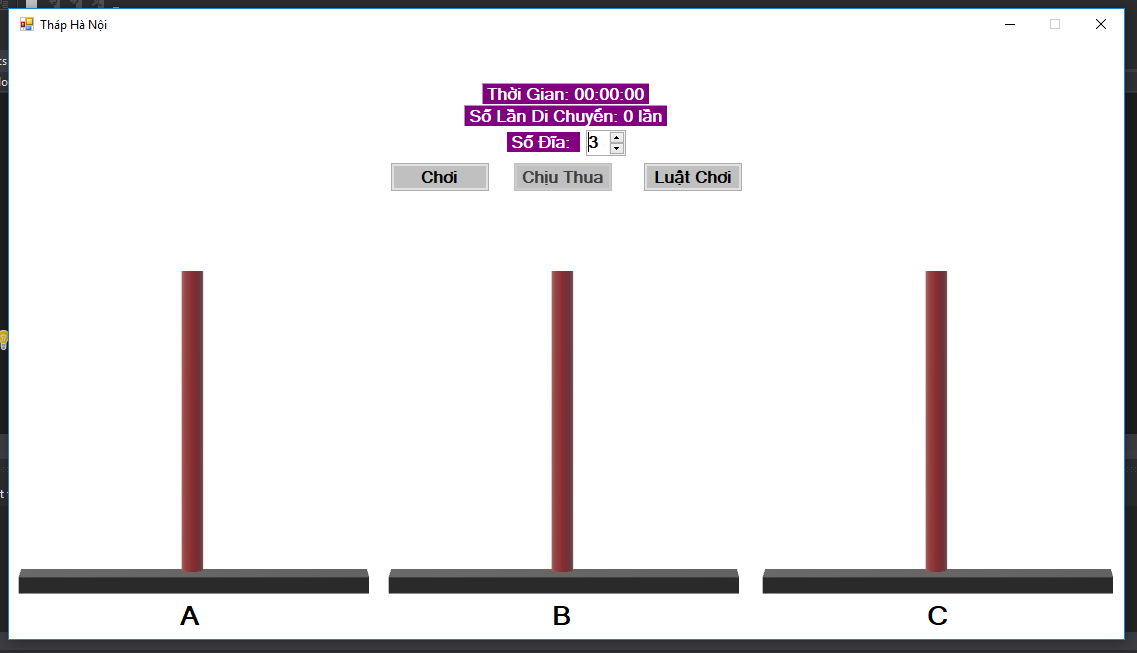
### Hàm di chuyển đĩa

* Cập nhật thông tin đĩa đầu bằng cách lấy thông tin và xóa đĩa ở cột đầu bằng *PictureBox firstTopDisk = firstClickedDisks.Pop().*
* Cập nhật mới tọa độ, tọa độ mới được truyền vào *firstTopDisk.Location = point.*
* Bỏ đĩa đã lấy ra vào đầu của cột đĩa ta đã chọn sau bằng cách sử dụng câu lệnh *secondClickedDisks.Push(firstTopDisk).*
* Tăng số lần di chuyển và cập nhật hiển thị ra giao diện.
* Trả giá trị 2 lần click về giá trị *null*.
* Ẩn hết các viền của các cột (*BorderStyle.None).*
* Tiến hành kiểm tra chiến thắng. Nếu số đĩa bằng level người chơi chọn thì chiến thắng, khi đó chúng ta sẽ gọi lại nút “Chịu Thua” và hiển thị thông báo chiến thắng bằng *MessageBox.Show().*

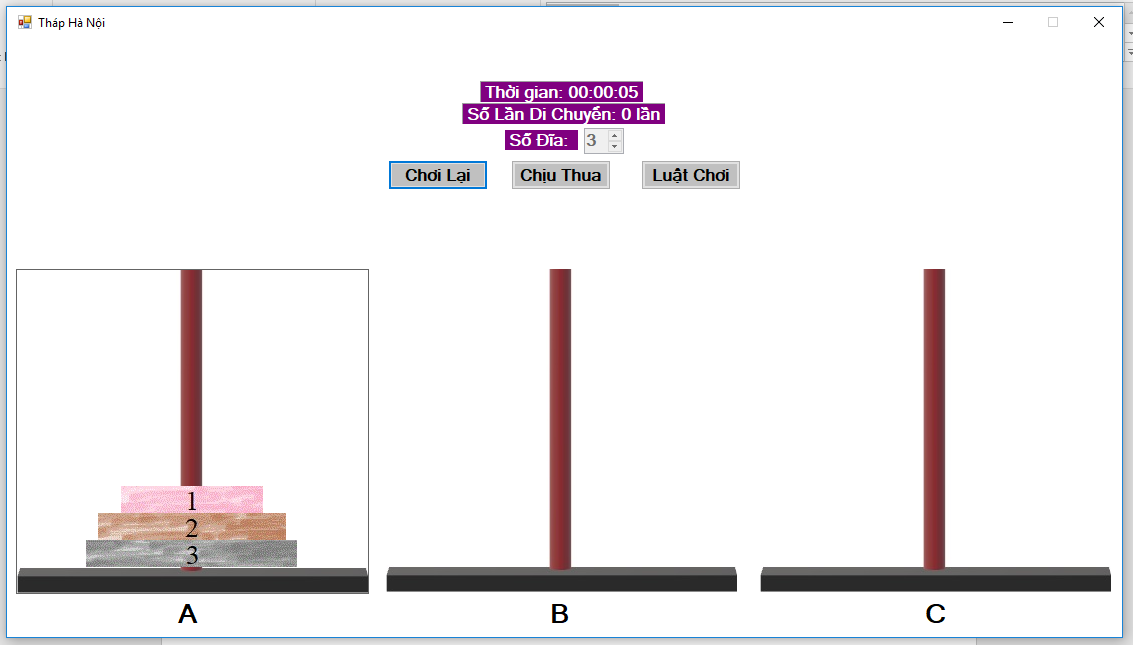


Hình 11: Hàm xử lí di chuyển đĩa.

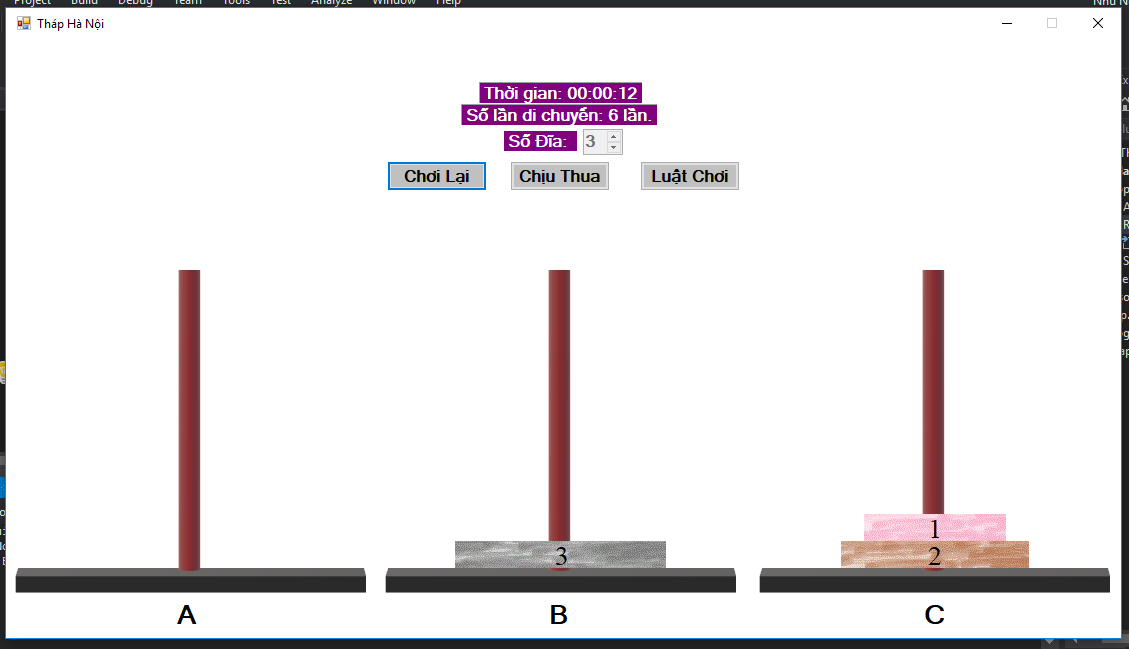
## Chạy và kiểm thử

Mở chạy thử chương trình sau đó tiến hành chọn chế độ chơi và bắt đầu chơi thử.

Hình 12: Trò chơi Tháp Hà Nội khi vừa mở lên.

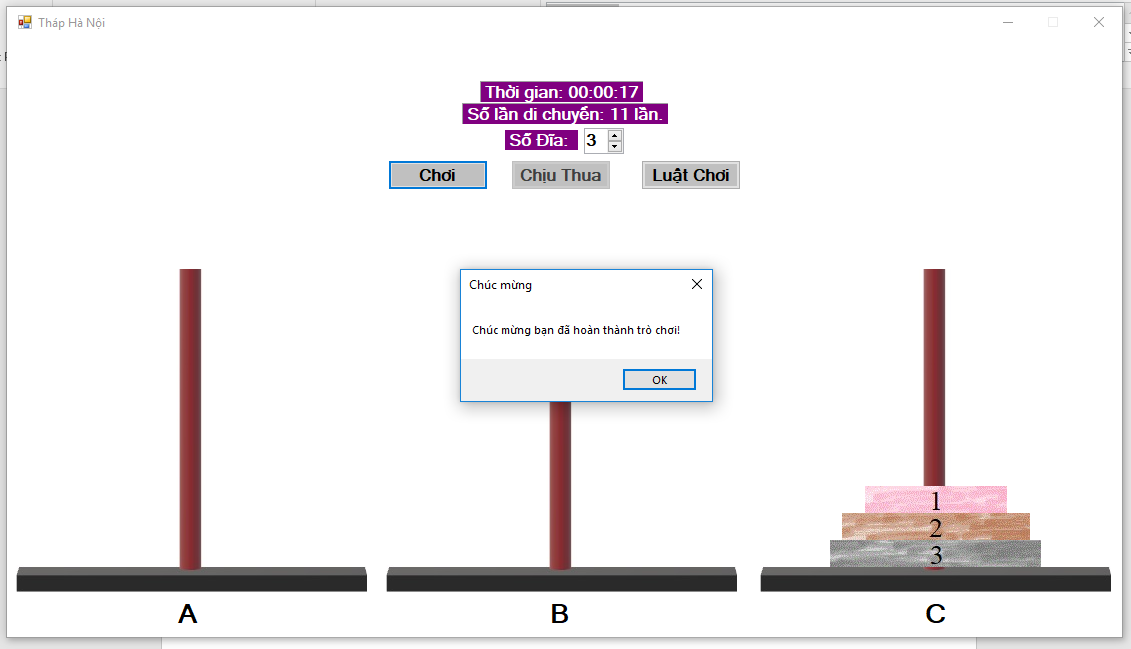


Hình 13: Trò chơi Tháp Hà Nội khi vừa chọn chế độ chơi.

**

Hình 14: Tiến hành di chuyển đĩa.

Hoàn thành trò chơi.



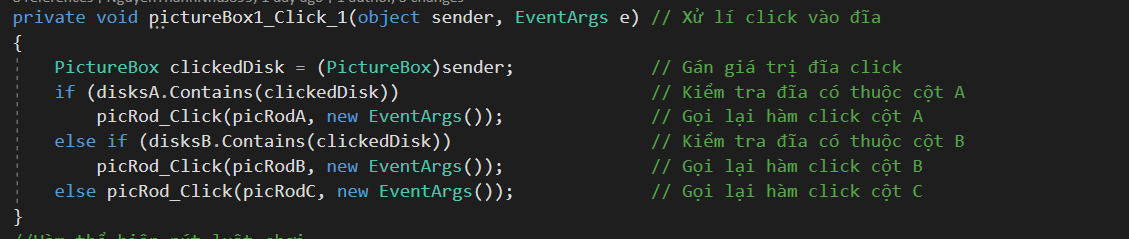
Hình 15: Thông báo chúc mừng khi người chơi hoàn thành trò chơi.

Tuy nhiên trong quá trình chạy thử đã xảy ra lỗi khi click trực tiếp vào đĩa thì cột sẽ không hiển thị. Tiếp theo chúng ta sẽ tìm hiểu và sửa lỗi.

## Sửa các lỗi gặp phải

Như đã nêu ở trên trong quá trình chạy thử đã xảy ra lỗi, không thực hiện được thao tác click khi ta click trực tiếp vào đĩa và cột. Sau khi tìm hiểu thì nguyên nhân chính là PictureBox của các đĩa đã nằm chồng lên PictureBox của các cột nên khi thao tác ta click hàm xử lí click cột sẽ không được xử lí dẫn đến xảy ra lỗi.

Sau đây chúng ta sẽ tiến hành sửa lỗi bằng cách viết hàm xử lí click đĩa. Đầu tiên ta tiến hành lấy giá trị đĩa đã click sau đó thực hiện kiểm tra với từng cột, nếu đĩa thuộc cột nào ta tiến hành gọi lại hàm xử lí click cột đó.



Hình 16: Hàm xử lí click đĩa.

# Mô tả phân công công việc

|  |  |  |
| --- | --- | --- |
| Tên SV | Đánh giá chung phần trăm đóng góp | Mô tả khái quát mảng công việc SV thực hiện trong đồ án. |
| Nguyễn Thành Như | 100% | Thực hiện toàn bộ đồ án. |

Bảng mô tả phân công công việc.

# Kết luận

* Mức độ hoàn thành mục tiêu: 100%.
* Các khó khăn gặp phải: chưa biết cách sử dụng Winform, việc tạo chuẩn bị hình ảnh để đưa vào giao diện chưa phù hợp dẫn đến phải sửa nhiều lần mới có size ảnh phù hợp.
* Cách khắc phục: tìm hiểu và học cách sử dụng Winform từ nhiều nguồn tài liệu trên internet, điều chỉnh size ảnh thích hợp cho việc code cũng như dễ nhìn cho giao diện.
* Ưu điểm của đồ án: sử dụng, khai thác hết các thuộc tính của kiểu dữ liệu Ngăn xếp (Stack), có tính ứng dụng trong giải trí và cả giáo dục phát triển tư duy logic, giao diện dễ nhìn, thân thiện, người chơi có thể dễ dàng thực hiện trò chơi.
* Những khuyết điểm của đồ án: so với các sản phẩm tương tự khác bên ngoài thì sản phẩm hiện tại vẫn còn chưa phát triển hoàn thiện nhất về các tính năng (có thể thêm các tính năng mới), giao diện chưa thật sự quá bắt mắt, cầu kì bằng các sản phẩm khác.
* Hướng phát triển đồ án: có thể thêm các tính năng khác như thêm một label hiển thị số lần di chuyển ít nhất số đĩa để chiến thắng, quy định số lần di chuyển nhiều nhất, nếu vượt quá sẽ thông báo người chơi thua, tương tự như vậy ta cũng có thể thêm thời gian chơi tối đa của mỗi level chơi. Ngoài ra chúng ta có thể thêm âm thanh cho mỗi lần click chuột (hợp lệ và không hợp lệ sẽ có âm thanh khác nhau) và cả khi chiến thắng hay thua cuộc. Đầu tư thêm về phần xử lí đồ họa giúp tăng độ thẩm mỹ, bắt mắt thu hút người chơi hơn.

# Tài liệu tham khảo

[1] “LUẬN VĂN THẠC SĨ TOÁN HỌC: THUẬT TOÁN FRAME – STEWART GIẢI BÀI TOÁN THÁP HÀ NỘI TỔNG QUÁT” của tác giả: Nguyễn Thị Hồng Phượng, số hóa bởi Trung tâm Học liệu – Đại học Thái Nguyên.

[2] “GIÁO TRÌNH: CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT” của tác giả: Lê Văn Vinh, NXB Đại học Quốc gia TP. Hồ Chí Minh.

Tài liệu tham khảo cho đề tài: Video hướng dẫn “Lập Trình C#\_Tháp Hà Nội\_ Sử dụng Stack để xây dựng” của tác giả: Lưu Thế Phương, Trung tâm Tin Học Đại học KHTN TP. Hồ Chí Minh.